In the Specification

Please delete paragraph [0005] in its entirety and replace it with the following:

[0005] Building a modern software application often is a complicated process. The application or software may comprise multiple executable tasks, modules, or components as well as various data and/or configuration files. Parts of the software may need to be created from computer program source files by compiling the computer programs into object code or into byte code and then linking the object code into an executable (byte code may be run directly by an interpreter and need not be linked to be made executable). Compiling source code may need to take place in a particular order to resolve dependencies between the source code files. In addition to compiling source code, the building process may involve validating either the presence or contents of configuration files and also may involve processing other kinds of files or data. The end products of software build activities may be stored in archive files, for example, in ~~Java~~ JAVA archive (JAR) files. The number of sources and other files that are needed to build an application may be quite large, which adds to the complexity of building the application.

Please delete paragraph [0007] in its entirety and replace it with the following:

[0007] A code control system (CCS) is commonly associated with a software build system. The CCS provides mechanisms for controlling changes to the various files and sources which are employed to build the software as well as for controlling the resultant

products of the build process. One of these mechanisms may include a locked check-out which prevents two developers from corrupting each other's work. Without a locked check-out mechanism, for example, if two developers check-out copies of the same file, each developer changes their checked out copy of the same file, each developer checks in their edited copy of the file, the last copy to be checked in writes over the edits of the first copy to be checked in, destroying those edits. With a locked check-out mechanism, only one developer is permitted to check-out a file for writing at a time. Others may check-out readable copies but not editable copies of the locked file. Another mechanism may enforce a policy that files may not be checked-out for writing without providing a reference to an active bug report or authorized software change request. Another mechanism may enforce a policy that files may not be checked-in without providing a reference to results of testing the changed software. A CCS may be purchased or leased as an off-the-shelf software product, for example ~~Merant's PVCS~~ MERANT's PVCS version management system or IBM's ~~Rational ClearCase~~ RATIONAL CLEARCASE software configuration management system.

Please delete paragraph [0020] in its entirety and replace it with the following:

[0020] The software build tool of the present disclosure combines a ~~Java-based~~ JAVA-based website interface and Jakarta ANT scripts working in coordination with the ~~Merant PVCS~~ MERANT PVCS version management system to manage the creation of Workflow Broker build ~~java~~ JAVA archives (JARs). One objective for the software build tool is to

provide a unified tool for introducing changes of JARs, for approving changes to JARs, and for building JARs.

Please delete paragraph [0021] in its entirety and replace it with the following:

[0021] Turning now to Figure 1, a block diagram of a software build tool system 10 is depicted. A code control system (CCS) 12 stores the code sources, various files, and other input which are processed to build a software product or application. The code sources may include files in various programming languages, for example, ~~Java~~ JAVA and C++. The various files may include, for example, configuration text files, interface definition language (IDL) files, and data type definition (DTD) documents. An IDL file contains a language independent definition of how two modules or systems communicate with each other. When two systems communicate by passing text files back and forth between each other, for example by passing extensible markup language (XML) text files, a ~~BusinessWare~~ BUSINESSWARE application must be informed of what these text files contain. ~~BusinessWare~~ BUSINESSWARE applications obtain this information through examining metadata contained in metadata classes. The DTD documents may be compiled to produce the needed metadata classes, as described further hereinafter.

Please delete paragraph [0022] in its entirety and replace it with the following:

[0022] The CCS 12 also stores the intermediate and end products of the software build process including object files, compiled byte code files, executable files, and Workflow Broker build JAR files. The Workflow Broker may be a component or application of an

commercial off-the-shelf (COTS) application such as VITRIA's BUSINESSWAREVitria's Business Ware, which has been customized or tailored for these purposes. The CCS 12 also stores files associated with the mechanics of the build process itself including script files and a change report 22. The CCS 12 may be an off-the-shelf software application, for example the MERANT PVCS Merant PVCS version management system. The CCS 12 may be associated with storage 14 where source files, configuration files, object code files, linked files, and other build product data are physically stored. In one embodiment, the CCS 12 provides the preferred method to access the storage 14.

Please delete paragraph [0025] in its entirety and replace it with the following:

[0025] When a software build is required, for example, when a software module is changed, an event is generated which causes a parser module 20 to parse the standardized comments associated with the designated changed files and generates from these comments a change report 22, also referred to as a change matrix, which consolidates, for example in one document or file, the information necessary to define or specify the software build to be performed. The change report 22 may contain entries for multiple changed files, wherein one entry in the change report is associated with each changed file. In some embodiments, the Java JAVA properties format may be used to parse comments. In one embodiment, the change report 22 may be captured in a Microsoft Excel MICROSOFT EXCEL spreadsheet document, but the software build tool system 10 is not limited to employing a Microsoft Excel MICROSOFT EXCEL spreadsheet document to represent the change report 22. The change report 22 is

stored in the CCS 12. The event which triggers the parser module 20 to generate the change report 22 may take different forms in different embodiments. For example, in a PVCS version management system the event may be the developer changing the status of an archive from SUBMIT status to CHANGED status, which is also known as promoting from SUBMIT status to CHANGED status. In other embodiments, the generation of the change report 22 triggers an email to be sent to build administrators that states that a build is waiting for their attention.

Please delete paragraph [0028] in its entirety and replace it with the following:

[0028] Several off-the-shelf compiler programs may be invoked if the source code is in more than one programming language, because off-the-shelf compiler programs are language specific. The compiler programs typically produce an intermediate object file from each source code file. ~~Java~~ JAVA compilers, however, typically compile Java source code files into byte code files which are ready to run on a ~~Java~~ JAVA virtual machine interpreter without any further processing. The object files and byte code files produced by the off-the-shelf compiler programs are stored in the CCS 12.

Please delete paragraph [0029] in its entirety and replace it with the following:

[0029] The next phase of the software build process involves a linker module 26 linking object files into executable files. The linker module 26 may comprise one or more scripts which invoke an off-the-shelf linker program. The linker module 26 has access to the object files via the CCS 12. The off-the-shelf linker program produces executable files

from the object files. In some embodiments there may be no object files produced, for example, in a 100% ~~Java~~ JAVA based source code system, and hence in this embodiment there would be no need for a linker module 26.

Please delete paragraph [0033] in its entirety and replace it with the following:

[0033] Turning now to Figure 2, a block diagram of another embodiment of a software build tool system 27 is depicted. A user interface 28 allows users to check-out and check-in code files and other files from the CCS 12. In this embodiment, the CCS 12 is a ~~Merant PVCS~~MERANT PVCS version management system, but in other embodiments a different off-the-shelf CCS 12 may be employed. A code compiler module 30 comprises one or more off-the-shelf compiler programs. A document type definition (DTD) compiler 32 compiles DTD documents into metadata classes. In the preferred embodiment, the metadata classes support VITRIA's BUSINESSWARE~~Vitria's BusinessWare~~ off-the-shelf software. An interface definition language (IDL) grinder 34 transforms IDL files into code stubs. The grinder 34 transforms IDL files into Java code stubs. Grinding is the interim act of compilation that is necessary to generate dependent class files for the host application to import required data sets(s). The subsequent importing and registering of these data sets with the host application, ensures a common interface definition persists between applications/technologies. IDL stub compilation is based upon the language, for example ~~Java~~ JAVA or C++, and creates a class file which then becomes the basis for subsequent code, such as ~~Java~~ JAVA or C++, to be compiled against.

Please delete paragraph [0034] in its entirety and replace it with the following:

[0034] The software build tool system 27 includes the user web client 16 interface and the administrator web client 18 interface. These web clients are supported by a web site application constructed as a standard ~~Java~~ JAVA Servlet/~~Java~~ JAVA Server Pages (JSP) 2.3 web application. This web application is based on the Jakarta Struts web application framework. Struts is a simple web framework that provides a clean organization of the parts of a Servlet/JSP website.

Please delete paragraph [0035] in its entirety and replace it with the following:

[0035] The user logs on to the software build tool system 27 and triggers the parser 20 to generate the change report 22. In some embodiments, the ~~Java~~ JAVA properties format may be used to parse comments. The change report 22 may contain entries for multiple changed files, one entry per changed file. The user may review the change report 22 and submit the change report 22 for evaluation by the administrator. In this embodiment, the change report 22 is a MICROSOFT EXCEL ~~Microsoft Excel~~ spreadsheet file, but in other embodiments the change report 22 may be stored in a different file format. The software build tool system 27 may send an email containing a copy of the change report 22 in hypertext markup language format and containing a copy of a testing results document to the administrators.

Please delete paragraph [0036] in its entirety and replace it with the following:

[0036] The administrator logs on to the software build tool system 27 via the administrator web client interface 18 and reviews the change report 22. If the administrator decides that the software build should be launched, the administrator approves the change report 22 which triggers the software build to begin. In this embodiment, the administrator approves the change report 22 by changing the status of the change report 22 from the CHANGED status to APPROVED status, also referred to as promoting from CHANGED status to APPROVED status, in the ~~Merant PVCS~~ MERANT PVCS version management system, CCS 12.

Please delete paragraph [0047] in its entirety and replace it with the following:

[0047] The process proceeds to block 62f in which code sources are compiled. Only source files which have changed are compiled, assuring a short build time. Note that the IDL stubs are compiled before the source is compiled because the source code may depend upon the IDL stubs. Where ~~BusinessWare~~ BUSINESSWARE is used, the ~~BusinessWare~~ BUSINESSWARE Connector def classes are called to create the customer connector INIS. INIS may refer to any structure where all the property files are stored, such that these property files or configuration files provide environment specific information based upon where they are deployed. The classes to call are stored in the project configuration file. The build is associated with a timestamp, and the lock on the special file is released.

Please delete paragraph [0052] in its entirety and replace it with the following:

[0052] The embodiments of software build tool systems 10 and 27 described above may be implemented on any general-purpose computer with sufficient processing power, memory resources, and network throughput capability to handle the necessary workload placed upon it. Figure 5 illustrates a typical, general-purpose computer system suitable for implementing one or more embodiments disclosed herein. The computer system 380 includes a processor 382 (which may be referred to as a central processor unit or CPU) that is in communication with memory devices including secondary storage 384, read only memory (ROM) 386, random access memory (RAM) 388, input/output (I/O) [[390]] devices 390, and network connectivity devices 392. The processor may be implemented as one or more CPU chips.

Please delete paragraph [0054] in its entirety and replace it with the following:

[0054] I/O [[390]] devices 390 may include printers, video monitors, keyboards, mice, track balls, voice recognizers, card readers, paper tape readers, or other well-known input devices. The network connectivity devices 392 may take the form of modems, modem banks, ethernet cards, token ring cards, fiber distributed data interface (FDDI) cards, and other well-known network devices. These network connectivity [[392]] devices 392 may enable the processor 382 to communicate with an Internet or one or more intranets. With such a network connection, it is contemplated that the processor 382 might receive information from the network, or might output information to the network in the course of performing the above-described method steps. Such

information, which is often represented as a sequence of instructions to be executed using processor 382, may be received from and outputted to the network, for example, in the form of a computer data signal embodied in a carrier wave.